Journal of The Korea Institute of Information Security & Cryptology ISSN 1598-3986(Print)
VOL.28, NO.3, Jun. 2018 ISSN 2288-2715(Online)
https://doi.org/10.13089/JKIISC.2018.28.3.617

머신러닝을 이용한 권한 기반 안드로이드 악성코드 탐지*

강 성 은,† 응웬부렁, 정 수 환‡
숭실대학교

Android Malware Detection Using Permission-Based Machine Learning Approach*

Seongeun Kang,† Nguyen Vu Long, Souhwan Jung‡
Soongsil University

요 약

본 연구는 안드로이드 정적분석을 기반으로 추출된 AndroidManifest 권한 특징을 통해 악성코드를 탐지하고자 한다. 특징들은 AndroidManifest의 권한을 기반으로 분석에 대한 자원과 시간을 줄였다. 악성코드 탐지 모델은 1500개의 정상어플리케이션과 500개의 악성코드들을 학습한 SVM(support vector machine), NB(Naive Bayes), GBC(Gradient Boosting Classifier), Logistic Regression 모델로 구성하여 98%의 탐지율을 기록했다. 또한, 악성앱 패밀리 식별은 알고리즘 SVM과 GPC (Gaussian Process Classifier), GBC를 이용하여 multi-classifiers모델을 구현하였다. 학습된 패밀리 식별 머신러닝 모델은 악성코드패밀리를 92% 분류했다.

ABSTRACT

This study focuses on detection of malicious code through AndroidManifest permissoion feature extracted based on Android static analysis. Features are built on the permissions of AndroidManifest, which can save resources and time for analysis. Malicious app detection model consisted of SVM (support vector machine), NB (Naive Bayes), Gradient Boosting Classifier (GBC) and Logistic Regression model which learned 1,500 normal apps and 500 malicious apps and 98% detection rate. In addition, malicious app family identification is implemented by multi-classifiers model using algorithm SVM, GPC (Gaussian Process Classifier) and GBC (Gradient Boosting Classifier). The learned family identification machine learning model identified 92% of malicious app families.

Keywords: android, malware, static analysis, machine leaning

I. 서 론

2013년 스마트폰의 보급률은 73%를 넘어서, 인구당 스마트폰 보급률은 2014년 기준으로 24.5%로 PC 보급률을 넘어섰으며, IDC에서 분석한 2016년 3분기 스마트폰 OS 시장 점유율은 안드로이드가 86.8%로 가장 점유율이 높은 것으로 나타났다[1][2]. 스마트폰의 보급률이 증가함에 따라 스마트폰은 전화 통화, 은행, 메신저 등 다양한 서비스를 제공하고 있으며, 사용자의 중요한 데이터들이 보안 위

Received(03. 22. 2018), Modified(06. 04. 2018), Accepted (06. 05. 2018)
* 이 논문은 2018년도 정부(미래창조과학부)의 재원으로 정보통신기술진흥센터의 지원을 받아 수행된 연구임 (No.2016-0-00078, 맞춤형 보안서비스 제공을 위한 클라우드 기반 지능형 보안 기술 개발). 또한, 이 논문은 2018년도 정부(과학기술정보통신부)의 재원으로 정보통신기술진흥센터의 지원을 받아 수행된 연구임 (No.2017-0-01853, 머신러닝 기반 지능형 악성코드 분석 통합 플랫폼)
† 주저자, y7k901@gmail.com
‡ 교신저자, souhwanj@ssu.ac.kr(Corresponding author)

협에 노출되고 있다. McAfee의 2017년 보고서에 의하면 모바일 악성코드는 2015년 4분기 이후부터 계속적으로 증가하여 2억개 이상의 모바일 악성코드가 발견되었다. 이중 새로운 형태의 악성코드는 2017년 2분기에 비해 3분기에 60% 급등하였다. 이는 새로운 악성코드에 대한 신속한 탐지 및 대응책이 필요하다[3].

본 연구는 이러한 급증하고 있는 안드로이드 악성코드를 빠른 시간 안에 정적분석을 기반 머신러닝 모델을 구축하여 악성코드를 탐지하고 리패키징을 통한 악성코드 패밀리 식별을 목표로 한다.

II. 관련연구

2.1 APK 분석

안드로이드에서 정적분석은 APK파일을 통해 수행되며 APK파일 내부에는 AndroidManifest.xml, lib, res, Dex 파일등 어플리케이션이 구동될 때 필요한 파일들을 내포하고 있다. APK내부 파일을 이용해서 API, 권한, 인텐트 정보들을 추출 할 수 있고, 파일 확장자와 헤더 확장자가 다른 위조된 파일 정보 및 어플리케이션 서명 정보도 추출 할 수 있다. APK는 Fig.1과 같이 7개의 폴더 및 파일로 구성된다.

- **classes.dex**는 안드로이드 Dalvik 가상머신이 인식할 수 있도록 class 파일을 모두 취합하여 바이트 코드로 변환한 파일이다.
- **res**는 컴파일 되지 않은 이미지 및 xml 리소스 파일들이 존재하는 폴더이다.
- **lib**는 라이브러리를 모아둔 폴더로 NDK (Native Develop Kit)으로 만들어진 각 프로세스에 알맞게 컴파일 된 so파일이 존재한다.
- **assets**은 AssetsManager에 의해 관리할 수 있는 어플리케이션의 정보를 포함하는 폴더이다.
- **META-INF**는 서명과 관련된 폴더이다. 내부에는 MANIFEST.MF, CERT.SF 등이 있으며 SHA1과 base64를 이용한 서명한 값을 저장하고 있다.
- **resources.arsc**는 리소스 파일들에 대한 정보를 기록한 파일이다. res폴더의 각종 리소스파일들의 type과 id 정보를 저장하고 있다.
- **AndroidManifest.xml**은 어플리케이션을 관리하기 위한 xml 파일이며 어플리케이션의 권한, 인텐트, 서비스, 액티비티와 같은 컴포넌트 정보, SDK 버전에 대한 정보를 명시하고 있다.

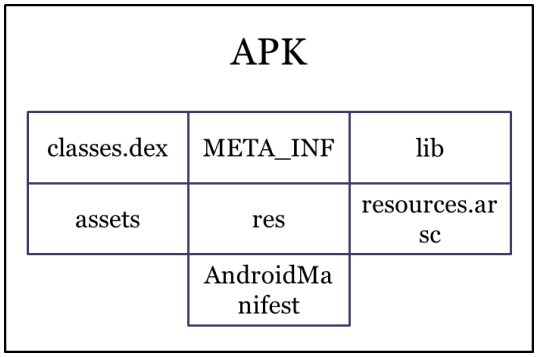

Fig. 1. internal Files in the APK

이러한 정적분석을 통해 Andro-AutoPsy Anti-malware 시스템은 인증서, 악성행위 관련 API, 권한, 인텐트 등의 정적분석 특징을 사용하여 99.45%라는 탐지 결과를 도출하였다. 하지만 악성 코드 패밀리 분류에서는 다소 저조한 결과를 얻었다[4]. 이 외에도 리소스와 파일 확장자, 헤더 magic값, assets와 lib 폴더들을 체크하는 탐지 방법도 있다[5].

2.2 Dex 분석

안드로이드 악성코드 분석을 위해서는 APK파일 내부의 AndroidManifest.xml과 Dex파일 분석이 중요하다. AndroidManifest.xml은 apk parse, aapt, apktool툴을 이용하여 정보를 추출이 가능하다[6][7]. 이러한 권한에 값을 설정한 후 리스크를

Table 1. Dex file structure

name	Description
string_ids	List of strings used within the app
type_ids	type identifiers list.
proto_ids	method prototype list
field_ids	class field list
method_ids	method identifiers list
class_defs	class definitions list.
call_site_ids	call site identifiers list.
method_handles	method handles list
data	The actual data area of the list areas
link_data	Data area of a statically linked file

계산하여 악성코드를 탐지할 수 있다[8]. Table 1과 같이 Dex파일 헤더는 크게 string, type, proto, field, method, class_def로 구성되어 있다. Dex 파일의 헤더에는 magic, checksum, signature 값이 있으며 각각의 영역의 offset 값들을 나타내고 있다[9]. 따라서 헤더 분석을 통해 offset 값을 참조하여 데이터를 접근할 수 있다. 이를 통해 string, class, method의 영역에 접근하여 API 및 API흐름까지 추출할 수 있다[10].

III. Feature selection

3.1 권한 추출

정적분석을 위해 약 2000개의 어플리케이션을 apk parse를 이용하여 특징들을 추출하였다. apk parse는 오픈소스 툴로서 APK파일에 대한 MD5 값, 파일 사이즈, SDK 버전 정보, 라이브러리 정보, 안드로이드 권한 및 컴포넌트 정보 등을 제공해 준다. Table 2와 같이 악성코드에서 많이 사용되고 있는 130여개의 권한을 특징으로 선택하였으며 APK파일에서 추출된 권한목록을 입력데이터로 활용하여 악성코드 탐지 및 악성코드 패밀리식별에 사용하였다.

Table 2. List of permissions used in Feature

Permission name	Description
CALL_PHONE	Call Permissions
INSTALL_PACKAGES	Application install permission
INTERNET	Internet access permissions
READ_CONTACTS	contact read permission
READ_EXTERNAL_ STORAGE	SD card read permission
READ_SMS	SMS Read permission
RECEIVE_BOOT_ COMPLETED	Boot complete event permission

3.2 API 매칭

안드로이드 악성코드의 대부분은 민감한 권한을 AndroidManifest.xml에 무분별하게 사용하고 있다. 이는 권한으로만 악성코드를 탐지할 경우 민감한 권한을 사용하는 정상 어플리케이션도 탐지하여 오탐률을 늘리게 된다. 이러한 이유로 실제 악성코드의 Dex 파일을 파싱하여 매칭되는 API를 확인해야하며 사용되지 않은 권한 및 사용된 권한에 대한 식별을 통해 악성코드 패밀리의 행동들의 구분이 필요하다. 3.1장에서 언급한 130여개의 권한과 관련된 API를 정리했으며 Table 3은 권한별 민감한 API매칭의 일부분을 보여준다. 관련된 API들은 Dex 파일 파싱을 통해 클래스 및 메소드의 문자열 값을 추출하였으며 매칭 결과에 따라 사용된 권한과 사용되지 않은 권한을 구분하여 특징으로 사용하였다.

Table 3. API Classification by Permission

Permission	API
ACCESS COARSE LOCATION	android.location.Address android.location.Criteria android.location.Geocoder android.location.GnssClock android.location.GnssMeasurement ….
DEVICE ADMIN ENABLED	android.app.admin.ConnectEvent android.app.admin.DeviceAdminInfo android.app.admin. DeviceAdminReceiver android.app.admin. DeviceAdminService ….
ACCESS KEYGUARDSECURE STORAGE	LockSettingsService.setLockPassword LockSettingsService.checkPassword LockSettingsService.checkPattern ….
BATTERY STATS	BatteryStatsService. getAwakeTimeBattery BatteryStatsService. getStatistics BatteryStatsService. getStatisticsStream ….

IV. 구 현

4.1 API 구현

본 연구에서 구현된 API는 총 3가지로 구현되었으며, preparing_data 함수는 어플리케이션의 데이터를 추출하여 몽고 DB에 Json 형태로 저장하는 함수이다. measure_accuracy는 저장된 특징들을 통

해 머신러닝에 학습시키고 테스팅하는 함수이다. 마지막으로 classify_family는 분류된 악성코드 패밀리를 도출하는 함수이다. Table 4와 Table 5는 각각의 함수의 내부적인 함수의 기능을 나타내고 있으며 API들은 scikit-learn 라이브러리를 통해 구현되었다[11].

Table 4. measure_accuracy function

measure_accuracy	
fit_the_4models	validate_models
This function generates four models with training sets. (SVM, NB, GBC, LG)	This function provides the agreed detection result of the models generated from fit_the_4models

Table 5. classify_family function

classify_family	
fit_the_4models_2	classify_malware_family
This function generates four multi- classifiers models with training sets. (SVM, GBC, GPC 1vs1, GPC 1vsRest)	This function provides the agreed identification result of the model generated from the fit_the 4models_2

4.2 분석 과정

본 연구에서 제안하는 모델은 악성코드 탐지부분과 악성코드 패밀리 분류 부분으로 구성되며 악성코드 탐지 부분은 SVM, NB, GBC, Logistic Regression로 구성되었다. 마찬가지로 패밀리 식별은 SVM, GBC, GPC 1vs1, GPC 1vsRest multi- classifiers 모델로 구축되었다. Fig. 2의 분석과정은 다음과 같다.

1. APK내부의 AndroidManifest으로부터 사용된 권한들을 추출하고, Table 3의 권한별 API들의 목록들을 Dex파일에서 Table 1의 method_ids와 class_defs로부터 지정된 클래스와 메소드들을 확인한다.
2. 확인된 권한들은 머신러닝에 사용되기 위해서 사용 및 사용되지 않은 권한으로 구분하여 저장되어지며, 매칭된 API도 몽고DB에 트레이닝 세트로 저장되어 진다.
3. 각각의 모델들은 저장된 트레이닝 세트를 통해서 학습을 실시하며, 총 8가지의 모델들이 악성코드 탐지와 악성코드 패밀리 탐지로 나뉘어 생성된다.
4. 새로운 APK파일이 주어지면 1,2 과정과 동일하게 특징을 추출하여 테스팅 세트에 저장된다. 생성된 모델들은 저장된 특징을 바탕으로 악성코드 탐지 및 악성코드 패밀리 구분을 실시한다.
5. 악성코드 탐지 및 패밀리 구분 모델들은 도출된 결과가 일치 하지 않을 경우 2개 이상의 모델들의 동일한 탐지 값을 결과로 합의한다.

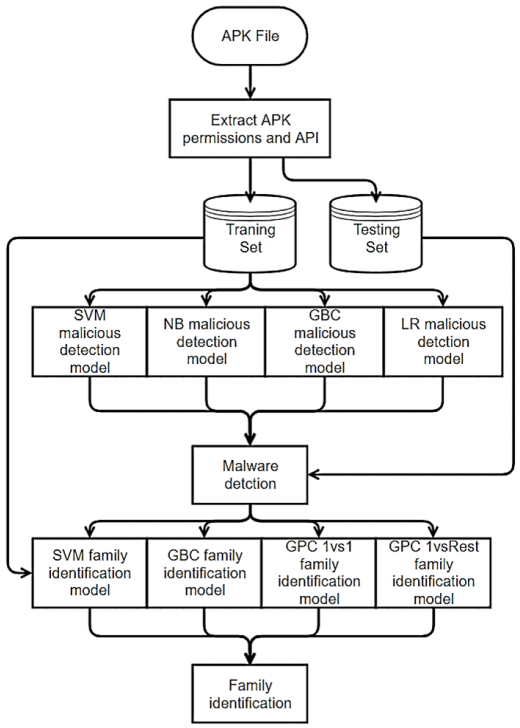

Fig. 2. Analysis process

기존 정적분석기반 악성코드 탐지연구에서는 권한 및 API에 대한 상세한 매칭이 없이 단일 특징 벡터로 악성코드 탐지를 실시하였다[12]. 권한 및 API를 기반으로 한 SVM, Bagging, J48 모델들을 통해 악성코드 탐지를 실시하였으나 권한과 API 매칭이 없이 각각의 모델을 구축하여 통합된 결과를 얻지 못하였다. 또한, 기존의 연구에서 동일한 특징을 사용하여도 머신러닝 알고리즘과 트레이닝 및 테스팅 세트에 따라 다른 결과를 보이고 있다[13]. 하지만 본 연구에서는 악성코드에서 사용된 권한 분류 작업을 통해 실제 코드에서 실행된 권한을 새로운 특징으로 포함하고 각각의 모델들의 결과를 합의하여 최종 결과로 도출하기 때문에 머신러닝 알고리즘에 대한 의존도를 낮췄다.

V. 실 험

본 실험에 사용된 서버의 성능은 Table 6과 같으며, 실험에 사용된 데이터는 정보보호 R&D 데이터 챌린지의 어플리케이션을 활용했다. 정상어플리케이션 1500개 악성코드 500개의 트레이닝 세트를 통해 머신러닝 모델을 구축하였다. 마찬가지로 테스팅 세트 1500개의 정상어플리케이션과 500개의 악성코드로 구성되었다. 추출된 특징들을 트레이닝한 시간은 평균 10.952초이며, 악성코드를 탐지한 시간은 55.529초로 총 66.481초의 시간이 걸렸다. Table 7과 같이 실험된 테스팅 세트를 통해 True Negative 1491, True Positive 474로 98.25%라는 탐지 결과를 얻었다. Table 8은 테스팅 세트를 통해 악성코드를 탐지한 후 악성코드의 패밀리를 분류한 결과이며, 악성코드 패밀리를 구분한 결과 28개를 오탐하였고 정상어플리케이션은 130개를 오탐하여 92.1%라는 정확도를 얻었다.

Table 6. Analysis server environment

OS	Ubuntu 16.04 xenial 4.4.0-101-generic
CPU	Intel(R) Xeon(R) CPU E5-2630 v3 @ 2.40GHz
CPU Core	8
Total memory	32836644 kB
Free memory	4325872 kB
Data slot duration	27.136ms
Information slot duration	2.7136ms
Data slots per frame	10

Table 7. Malicious app detection result

Category		result	
		Malware	Benign
Experiment result	Malware	474	9
	Benign	26	1491

Table 8. Malicious family identification result

	Family name	Number of families	Experiment result	
			true	false
result	wapsx	50	43	7
	gappusin	50	42	8
	smstado	50	47	3
	adwo	50	50	0
	counter	50	50	0
	airpush	50	40	10
	dowgin	50	50	0
	opfake	50	50	0
	smsagent	50	50	0
	boxer	50	50	0

VI. 결 론

본 연구는 정적분석 권한 기반으로 머신러닝을 이용하여 악성코드를 분류하였다. 악성코드 탐지 및 패밀리 식별 결과 98%, 92%로 높은 수치를 얻었다. 또한, 구축된 모델을 통해 2000개를 테스팅 하는데 평균 66초라는 결과를 얻었다. 기존 연구에서 사용된 정적분석의 특징들에 더 나아가 실제 악성코드를 분석하여 사용되는 API와 권한들을 최적화하여 특징들을 적용하였으며, 이를 통해 정적분석에 대한 시간을 줄이면서 악성코드 탐지 및 패밀리 식별에서 높은 결과를 얻을 수 있었다.

본 연구에서 airpush 악성코드와 정상어플리케이션을 비교 분석한 결과 READ_EXTERNAL_STORAGE 및 INTERNET 권한을 통해 악성행위를 실시하여 정상어플리케이션과 다른 뚜렷한 특징이 없었다. API경우에도 Evironment.getExternalStrogeDiretory와 같이 정상어플리케이션과 비교되는 특징이 없어 오탐이 되었다. 이러한 오탐을 줄이기 위해서는 Socket, Http API는 내부에 포함된 URL 및 IP 값에 대한 분석, SEND_SMS 권한과 발신 메시지 내용 분석, Shared Preference API의 저장되는 변수 값에 대한 분석을 통해 정상어플리케이션과 다른 특징을 추출할 필요가 있다. 이에 따라 API 매칭뿐만 아니라 인자값 분석 및 동적 분석을 통해 실제로 실행된 악성행위에 대한 명확한 특징을 반영해야한다. 하지만 동적 분석의 경우 코드 커버리지 및 분석시간에 대한 대책이 필요할 것이다.

본 연구는 급증하는 악성코드에 대한 자동탐지 및 조기대응에 대한 하나의 방법이 될 수 있다. 추후연구로 클래스와 메소드에 대한 인자값 및 시퀀스 분석과 악성코드 난독화를 대응하기 위한 so파일 분석 및 동적 분석을 적용할 예정이다.

References

[1] "Mobile trends in 2015," KT Economic Management Institute, Jan. 2015

[2] IDC, "http://www.idc.com/promo/smartphone-market-share/vendor", Mar , 2017

〔3〕 "Threat Report", McAfee Labs, Dec. 2017
〔4〕 Jang, Jae-wook, et al. "Andro-autopsy: Anti-malware system based on similarity matching of malware and malware creator-centric information," Digital Investigation vol.14, pp.17-35. 2015
〔5〕 Suarez-Tangil, Guillermo, et al. "DroidSieve: Fast and accurate classification of obfuscated android malware," Proceedings of the Seventh ACM on Conference on Data and Application Security and Privacy. pp. 309-320, Mar. 2017.
〔6〕 apk_parse, "https://github.com/tdoly/apk_parse," 2018
〔7〕 Android-apktool: A tool for reengineering Android apk files. "https://ibotpeaches.github.io/Apktool/," 2018
〔8〕 Sarma, Bhaskar Pratim, et al. "Android permissions: a perspective combining risks and benefits," Proceedings of the 17th ACM symposium on Access Control Models and Technologies. pp. 13-22, Jun 2012.
〔9〕 Dalvik Executable format, https://source.android.com/devices/tech/dalvik/dex-format, 2018
〔10〕 Yang, Zhemin, and Min Yang. "Leakminer: Detect information leakage on android with static taint analysis," Software Engineering (WCSE), 2012 Third World Congress on, pp. 101-104, Nov, 2012.
〔11〕 Pedregosa, Fabian, et al. "Scikit-learn: Machine learning in Python," Journal of machine learning research, pp.2825-2830. 12. Oct, 2011
〔12〕 Peiravian, Naser, and Xingquan Zhu. "Machine learning for android malware detection using permission and api calls," Tools with Artificial Intelligence (ICTAI), 2013 IEEE 25th International Conference on, pp.300-305, Nov. 2013.
〔13〕 Ham, Hyo-Sik, and Mi-Jung Choi. "Analysis of android malware detection performance using machine learning classifiers," ICT Convergence (ICTC), 2013 International Conference on, pp.490-495, Oct, 2013.

〈저자소개〉

강 성 은 (Seongeun Kang) 학생회원
2016년 8월~현재: 숭실대학교 정보통신공학과 석사과정
〈관심분야〉 정보보호, 모바일 보안, 클라우드 보안

응웬부렁 (Long Nguyen-Vu) 학생회원
2016년 2월: 숭실대학교 정보통신공학과 석사
2016년 3월~현재: 숭실대학교 정보통신공학과 박사과정
〈관심분야〉 네트워크 보안, 모바일 보안, 클라우드 보안

정 수 환 (Souhwan Jung) 종신회원
1985년 2월: 서울대학교 전자공학과 졸업
1987년 2월: 서울대학교 전자공학과 석사
1996년 6월: University of Washington 박사
1988년~1991년: 한국통신 전임 연구원
1997년~현재: 숭실대학교 전자정보공학부 교수
〈관심분야〉 클라우드 보안, 모바일 보안, 네트워크 보안